An OSI Architecture for the Deep Space Network

W. R. Heuser
Advanced Information Systems Section

This article presents an Open Systems Interconnection (OSI) architecture developed for the Deep Space Network. An historical review is provided to establish the context for current United States Government policy on interprocessor communication standards. An introduction to the OSI architecture, its seven-layer approach, and an overview of application service entities are furnished as a tutorial. Finally, the results of a prototype system developed for monitor and control of a Deep Space Station are also presented.

I. Introduction

The Deep Space Network (DSN) has, from its inception, employed automation whenever possible to support its mission: telecommunications with spacecraft in deep space. Over the past 25 years, advances in computer automation have accounted for substantial increases in operational productivity. Today, the Deep Space Stations are operated with a fraction of the personnel required 20 years ago. The DSN has evolved into a distributed computer system with all of its elements interconnected. The software programs (called application programs) used to operate the DSN are critically dependent on the software and hardware that support the exchange of data among DSN computers.

The development, integration, management, and maintenance of large distributed computer software systems is a costly enterprise. Although the price of computer hardware has dropped significantly over the past decade, the cost of software development has increased. In the DSN, software has become a major cost element. For example, the Signal Processing Center (SPC) modification project is a multimillion dollar effort and almost 80 percent is earmarked for software development. The DSN is not alone in facing the financial burden of the software development effort. The proliferation of distributed systems is a challenge facing government and the private sector.

Over the past two years, the DSN's Information Systems Division has conducted research into a distributed computer software architecture for the DSN that offers a solution to the high cost of distributed systems. The Open Systems Interconnection (OSI) architecture is an approach to interprocessor communications developed by the International Organization for Standardization (ISO) and is the focus of that research effort. The reasons for this focus are

(1) The OSI architecture meets the functional requirements of the DSN.

(2) The OSI architecture is designed to decouple the application program from the communications process. With OSI standards, the DSN can purchase commercial communication services and focus its resources on specific DSN applications.

(3) The OSI architecture has been adopted by the United States Government as the standard for distributed government computer systems [1].

(4) The OSI architecture has been selected by the Consultative Committee for Space Data Systems (CCSDS) as the model for the international space community [2].

An OSI-based architecture is expected to reduce the cost of distributed computer systems by providing standard communication interfaces for application programs. Market forces are driving the development of commercial products that provide communication services in accordance with the international standards. Products that meet the standards provide interprocessor communications between computers from different vendors. Competition between vendors will drive costs down.

This article is the result of two years of extensive research. It is written to provide a general education on OSI as well as to present the results of the research and prototype effort. To accommodate a wide audience with different levels of knowledge, the article is composed of four independent sections to permit selective reading. The sections are

(1) Section II: An Historical Perspective and Government Policy. An historical perspective on interprocessor communications is presented to provide an understanding of how the technology and government policy evolved.

(2) Section III: The OSI Basic Reference Model and Application Entities. An overview of the OSI Basic Reference Model is presented with a description of each layer's function and a description of the current OSI application entities.

(3) Section IV: An OSI Architecture for the DSN. An overview of an OSI-based DSN architecture is described. The DSN's data flow is partitioned between several application entities, and areas for additional research are identified.

(4) Section V: A DSCC Monitor and Control Prototype. A detailed report on an OSI-based Monitor and Control architecture for DSN tracking stations is presented. The potential for major cost savings is analyzed and the results of a prototype effort are detailed.

This approach is necessary because, in order to understand where one is and where one is going, it is necessary to understand where one has been. To achieve a comprehensive solution, one must understand the overall approach and the total problem. Finally, no large-scale implementation should be initiated without a small-scale prototype to verify the approach.

II. An Historical Perspective and Government Policy

In the late 1960s, the Defense Advanced Research Projects Agency (DARPA) recognized not only the demand for direct computer data exchange but the necessity for industrial standards that would be independent of the computer manufacturer. The Department of Defense (DoD) was already experienced in the business of computer data exchange and the inability of systems built by different companies to communicate. An early example was the 1950s Nike Ajax air defense system and the North American early warning radar system. The air defense system was developed under Army supervision and the radar system was developed under the Air Force [3]. When the time came to exchange data for a coordinated defensive response, the two systems were incompatible. With this historical perspective, DARPA began a long-term research effort through university and industrial laboratories to develop standards for interprocessor communications.

The first steps involved simple systems with two computers from the same manufacturer, connected by a cable. Addressing and protocol issues arose with the addition of the third and then the fourth computer. Still more problems arose with the introduction of computers from different manufacturers. Multiple cables were replaced by the development of media access control technology, and slowly the term local area network (LAN) emerged from the laboratory to the commercial world. DARPA's efforts culminated in 1983 with the publication of two DoD standards for interprocessor communications: Transmission Control Protocol (TCP) [4] and Internet Protocol (IP) [5]. Today, work continues in the university and commercial arenas, expanding what has become known as the DoD suite of TCP/IP standards.

The effort to develop standards for the computer industry is by no means a DARPA-exclusive activity. A number of different organizations, such as the Consultative Committee for International Telegraph and Telephone (CCITT) and the Institute of Electrical and Electronics Engineers (IEEE), are engaged in the standards development effort. These organizations have provided a wide variety of standards for different telecommunications media and media access techniques. The American National Standards Institute (ANSI) has contributed standards for character encoding and computer languages.

A more comprehensive and general solution to the problems of interprocessor communications is under development by the International Organization for Standardization (ISO). ISO began its effort in the mid-1970s and has

built on the lessons learned by DARPA. Called Open Systems Interconnection (OSI), the ISO approach provides for rapid changes in computer hardware and telecommunications technology with an architecture that insulates the user program from the manufacturer-dependent communication services. In addition, ISO recognized that the cost of software development was exceeding the cost of computer hardware, a significant change from 20 years earlier. Consequently, the OSI architecture was also designed to allow software systems to transcend computer hardware changes and communication systems evolution.

The growing costs to the federal government of computer hardware and software prompted the Office of Management and Budget in 1984, through the Office of the Chief Executive, to commission a study by the National Academy of Sciences on the status of interprocessor communication standards. The report *Transport Protocols for Department of Defense Data Networks* was published by the National Research Council (NRC) in February 1985 [6], with the recommendation that the United States Department of Defense adopt the International Organization for Standardization's OSI as the basis for all interprocessor communications. The NRC committee made its recommendation based primarily on two considerations:

(1) The ISO and DoD protocols are basically equivalent at the transport level.

(2) The worldwide market demand for ISO protocols is far larger than the market for the DoD protocol suite.

In response, the Department of Defense adopted OSI protocols as costandards with the DoD standards, with plans to make them "the sole mandatory interoperable protocol suite" [7].

In 1987, the United States Congress passed the Computer Security Act [8], which established the National Bureau of Standards (NBS) as the sole government agency responsible for the development of computer standards. This legislation also created a new category of Federal Information Processing Standards (FIPS's) called compulsory standards and modified the federal property laws to impose compulsory standards on the procurement of all federal government property. In April 1987, the first draft of the Government Open Systems Interconnection Profile (GOSIP) was released for comment. In August 1988, the draft document became the first compulsory Federal Information Processing Standard—146 [1].

The motivations for GOSIP are clearly stated in its introduction. "In the past, vendor-specific implementations of data communications protocols led to isolated domains of information, very difficult and expensive to bridge." Through GOSIP, "... the government expects to realize significant savings through reducing duplicate circuits and wiring, training, custom software, workstations, and custom hardware interfaces."

The Department of Veterans Affairs (VA) provides an excellent example of why GOSIP is necessary. The VA is a very large government agency, spread around the world, with computer networks numbering in the hundreds. Given the size of the VA, it is impossible for Congress to allocate funds to replace all the VA networks with a proprietary computer network in a single budget appropriation. In addition, the competitive procurement process makes it impossible to guarantee a single vendor source spread over many procurement cycles. Consequently, a vendor-independent standard is the only mechanism available for the acquisition of network products and services that ensure interoperability, and the selection of OSI provides the United States Government with an internationally recognized standard supported by a wide variety of vendors [9].

The National Institute of Standards and Technology (NIST, formerly the National Bureau of Standards) is responsible for the development of OSI network services in the government. Through GOSIP, NIST is creating the government market for commercial OSI products. NIST sponsors quarterly meetings of the NIST Workshop for Implementors of OSI to assist industry in the development of these OSI products. In addition, NIST is coordinating the conformance testing effort to assist government agencies in determining the interoperability of commercial products.

III. The OSI Basic Reference Model and Application Entities

A. Basic Reference Model

The objective of the OSI architecture is a system where user programs or application programs can employ the resources of any processor in a network without concern for the communications process or the computer hardware. To achieve this objective, ISO adopted a layered approach based on the functional partitioning of the communications process. The Basic Reference Model [10] emerged as an international standard (ISO 7498) in 1984 and defined seven layers, the so-called seven layer cake (see Fig. 1). Each layer has been assigned a specific set of services and an associate set of protocols. The seven layers are summarized below [11,12].

1. The application layer. This layer provides the user program with an interface to an OSI system. In this case, the user is any computer program or application program that requires interprocessor communications. A number of common communication functions have been identified and grouped into so-called application entities with standard services and protocols. The application entities are the key to open systems and will be discussed in more detail later.

2. The presentation layer. This layer (ISO 8823/9576) provides a common representation of application data that is communicated between application entities. Common representation refers to the encoding of data and the order of bits and bytes exchanged between processors. For example, the exchange of data between a processor using American Standard Code for Information Interchange (ASCII) encoded characters and a processor using Extended Binary Coded Decimal Interchange Code (EBCDIC) encoded characters requires a data translation before the information can be utilized. Another example of a presentation issue is the exchange of data between a 32-bit/word computer processor and a 16-bit/word processor where the bit and byte ordering must be adjusted for the correct interpretation of the data.

3. The session layer. This layer (ISO 8327/9548) provides dialogue services for those functions that require the establishment of a connection and synchronization between any two machines prior to the exchange of information. This layer provides the "Are you there"–"Yes I am" exchange prior to the exchange of application data.

4. The transport layer. This layer (ISO 8073/8602) provides those services required for the reliable end-to-end transmission of data. The issues of error detection, error recovery, and multiplexing are network dependent, and the transport layer manipulates the underlying network services to provide the appropriate quality of service (QOS).

5. The network layer. This layer (ISO 8473) provides the networkwide (or internetwork) services required to transmit, route, and relay data between computers without regard for the communications medium. In networks composed of several segments connected with bridges or independent networks connected through gateways, the network layer provides the services and protocols necessary to deliver the data to its final destination.

6. The data-link layer. This layer (ISO 8802-x) provides support for the physical medium employed in the transmission of data. ISO has established standards for a wide variety of transport media including four types of local area networks: an Ethernet (ISO 8802-3), a token bus (ISO 8802-4), a token ring (ISO 8802-5), and a Fiber Distributed Data Interface (FDDI).

7. The physical layer. This layer is the physical communication medium that supports the transmission of bits and is concerned with the electrical interface to the equipment supporting the transmission.

The process of establishing ISO standards is a difficult and time-consuming effort that involves committees from all participating nations. A standard begins as a draft proposal (DP), moves to a Draft International Standard (DIS), and achieves International Standard (IS) status over a period of three to six years. The key standards for the lower six layers (presentation layer to physical layer) have been International Standards (IS's) for more than three years. Some of the application layer entities are International Standards and others continue under development [12].

B. Application Entities

Much of the standards effort is now focused on the application entities, which are the key to the OSI architecture. Using the appropriate application entity, standardized software functions provide access to the OSI services required for interprocessor communications. The application entities are summarized below.

1. The Manufacturing Message Specification entity. This application entity (MMS, ISO 9506) provides a set of services developed from the Manufacturing Automation Protocol (MAP) initiative at General Motors in the early 1980s. Designed as a process control standard, the MMS Standard achieved IS status in 1989 and is aimed at direct interprocessor communications among machines on a factory floor. In the factory environment, an assembly line can be composed of hundreds of machines performing complex, precision tasks all as a unit. If any element of the assembly line fails to perform or performs at less than optimum performance, the assembly line as a whole fails to perform. The interprocessor communication provided by MMS is designed to support this type of environment. The application of MMS to the DSN is discussed in detail later [13].

2. The File Transfer, Access and Management entity. This application entity (FTAM, ISO 8571) provides a service and protocol standard to access and manage files in an open system and was one of the first application entities to achieve IS status in 1988. Using FTAM, a user program can open, read, write, and close files on

another processor just as though it were manipulating a local file. FTAM also provides services for copying files and obtaining file directories from remote systems. The application of FTAM services might greatly simplify the problems associated with centralized data recording at the Deep Space Communication Complexes (DSCC's). For example, a telemetry processor can open a file for recording on a remote network file server at the beginning of a spacecraft support pass, write the telemetry data to that file, and close the file at the completion of the activity. Playback could be accomplished in a similar fashion using the FTAM Read services.

3. The Network Management entity. This entity (NM, ISO 9595/9596) developed as a result of the growing use of distributed systems and the lack of mechanisms to monitor and manage the communications resources of networks. Though network management is still in the draft proposal stage of the standards process, five service elements have been identified as the core of this application entity:

(1) Performance will be monitored at each layer of the seven-layer architecture, providing information on byte counts, time delays, data rates, and other statistical information related to the performance of the communication process.

(2) Error Reporting and Logging will detect errors and failures in the communications process.

(3) Security guards against unauthorized access.

(4) Configuration Management will provide for the allocation and assignment of communications resources.

(5) Accounting Management will provide audit services [14–16].

4. The Message Handling Systems Service entity. This entity (MHS, ISO 10021), unlike other application layer entities, is a collection of international standards that together form the basis for OSI electronic mail (the X.400 standards). Many of the standards under the umbrella of MHS are the result of a long collaboration between ISO and the CCITT standards effort. The distribution of DSN operational support messages is the result of an excellent match of functional requirements.

5. The Directory Services entity. This entity (DS, ISO 9594) is intended to provide a global interconnected directory for all types of OSI entities, individuals, distribution lists, application entities, and general agents using network communications services. An international standard since 1989, Directory Services provides address resolution based on logical processor names anywhere in the network.

6. The Remote Database Access entity. This entity (RDA) is a proposed standard stimulated by the expanding use of commercial database systems. Still in the draft proposal phase, RDA is intended to provide a set of standards for access to any "open systems" database through a set of standard functions and a standard Sequential Query Language (SQL).

7. The Virtual Terminal entity. This entity (VT, ISO 9040) is a standard for terminals and hosts to communicate across different networks without requiring that one side know the terminal characteristics handled by the other side. An International Standard since 1988, VT provides a generic set of terminal characteristics for communication, which can be mapped to local terminal characteristics for display.

8. The Job Transfer and Manipulation entity. This entity (JTM, ISO 8831/8832) is designed to support computer-to-computer communications for the purpose of performing work remotely. JTM developed as a spin-off of batch processing and provides the protocols necessary to transfer and perform jobs on processors in a distributed system.

One of the most important aspects of the layered architecture is the flexibility to select application layer services that can operate over a number of different data-link and physical layer transmission standards. For example, an organization can implement an application based on the Manufacturing Message Specification operating over an Ethernet (ISO 8802-3) and move to an FDDI as its data rate requirements change. This approach provides an evolutionary path for distributed systems while preserving the organization's software investment.

In addition, the OSI-layered architecture is open to expansion as new problems and technologies evolve. In the mid-1980s for example, the Massachusetts Institute of Technology developed X Windows to support a variety of computer terminals. The X Windows concept is now in the standards process and will emerge in the future as a companion standard to the Virtual Terminal application entity [17].

IV. An OSI Architecture for the DSN

An OSI architecture for the DSN would provide access to communication resources through a standard OSI application layer entity. The appropriate selection of the application entity is the key and is based on the type of data flow to be supported. A study of DSN data flow reveals two general categories of data flow:

(1) Real-time spacecraft data: acquired, recorded, and transmitted back to JPL to support flight projects (the end product of the DSN).

(2) Process control: the data exchange that supports all aspects of DSN operations.

Included in the category of real-time spacecraft data are subcategories such as the playback of recorded spacecraft data in nonreal time. Under the category of process control are subcategories including the transmission of support data and network control data. The partitioning of data flow into these two categories provides a starting point for the partitioning of the system.

Under the category of real-time spacecraft data, the DSN is responsible for recording the spacecraft data types at the DSCC's. In addition, the DSN is frequently required to relay the spacecraft data types back to JPL in near-real time. The services and protocols provided by FTAM will support the functional requirements for recording spacecraft data at the Deep Space Stations. The telemetry subsystems at the stations can use FTAM software to open a client–server relationship with a station data recording subsystem (commonly referred to as a network file server). The FTAM services F_OPEN, F_WRITE, and F_CLOSE would be used to open files on the station data recorder, write the spacecraft data to those files, and close the files at the completion of the spacecraft track. The playback of recorded spacecraft data would employ a similar approach using the F_READ FTAM service in place of the F_WRITE service. These services use reliable connection-oriented Transport Protocol Four (TP-4) [11].

The process control category is composed of a diverse list of data flow types:

(1) Monitor data flow throughout the DSN to inform operations personnel of the status of all operational elements. The monitor data report the position of antennas, the condition of receivers, the available communications facilities, and the values of many other detailed components.

(2) Control data flow to each of the diverse elements to effect change and allocate DSN resources to support specific activities. Anomalies detected at the subsystem level generate messages to alert operations personnel.

(3) Support data files are transferred throughout the DSN to provide configuration and control information at the subsystem level.

The diversity of data flow results in a partitioning of functional requirements across several OSI application entities.

(1) Station Monitor and Control. The data flow for monitor and control of the Deep Space Stations is supported with the services of the OSI MMS and is discussed in more detail in the next section.

(2) Communications Resource Management. The OSI Network Management Service (NMS) provides the tools necessary to manage the communication resources of an OSI-compatible network. The security element of network management provides the apparatus to extend access to the DSN to those functions that require DSN connection, while providing the necessary safeguards against unauthorized access.

Other elements of network management identify and automatically report communications performance, problems, and failures, and thus provide DSN operations with the tools necessary to maintain the flow of data. The allocation of DSN resources in support of specific missions is reported through the accounting element of network management, which provides detailed information on DSN support performance and feedback to the scheduling teams.

(3) Support and Command Files. The transfer of support and command data is supported with the services of FTAM.

(4) Resource Identification. The identification of resources within the system is handled through address resolution with OSI Directory Services (DS).

(5) Operational Message Services. Administrative messages supporting operations are distributed through OSI Message Handling Systems (MHS—better known as X.400 electronic mail).

The typical architecture for an OSI-based DSN subsystem is shown in Fig. 2. The OSI application entities operating in each subsystem will vary from subsystem to subsystem depending on functional requirements.

One troubling issue remains and requires further research and analysis: the real-time relay of spacecraft data from the Deep Space Stations to JPL. The DSN employs dedicated leased communication lines to support the flow of data between JPL and the Deep Space Stations. The leased lines are generally expensive, have limited capacity (bandwidth), and operate through satellites in geosynchronous orbit. Traditionally, NASA has developed elaborate protocols to support this communication facility. These protocols are connectionless and designed to maxi-

mize the spacecraft data delivered over the limited capacity lines. Error detection and recovery are handled at the equivalent of the ISO data-link layer.

OSI connection-oriented protocols have been implemented over satellite communication systems and operate effectively. However, these protocols employ transport-level error detection and recovery. Large delays are typically experienced in transport-level acknowledgments over satellite systems and have a major negative impact on data rates. Connectionless OSI protocols could be used, however, to provide a service similar to the present DSN system. At present, there is no OSI application entity based on connectionless protocols, but the standards are in place for connectionless presentation (ISO 9576), session (ISO 9548), and transport (ISO 8602) layers. Development of a real-time data delivery OSI application layer standard is needed. Such a standard would employ Link Level Control 3 (LLC3) under ISO 8802-2 to provide a reliable data-link service.

The most important aspect of this solution is the incorporation of the ISO 8473 network protocol which provides conformance to the CCSDS recommendations and the proposed NASA Communications Network (NASCOM) gateway services [2,18]. Commercial high-speed multiprotocol routers that support ISO 8473 would provide LAN-to-LAN interconnection over standard commercial communication services. These routers are inexpensive and the cost to interconnect DSN facilities and support its customers would be greatly reduced.

V. A DSCC Monitor and Control Prototype

The Monitor and Control System for the DSN DSCC's was selected for an OSI-based prototype system. The prototype activity has provided hands-on experience with OSI protocols and is expected to substantially reduce the risk of errors associated with future implementations. The selection of the Monitor and Control System for the prototype effort was the result of several factors:

(1) The station monitor and control requirements are well defined and documented.

(2) The functional requirements for station monitor and control and the service specification for the MMS protocol are a close match.

(3) Commercial software products built to the MMS standard are readily available.

(4) The software and hardware products required for a prototype are available within the constraints for the research effort.

(5) The anticipated replacement of the DSCC Monitor and Control (DMC) Subsystem computer hardware in the mid-1990s provides an opportunity to transfer OSI technology to the DSN with substantial cost savings.

The present Mark IVa Monitor and Control System is a distributed software architecture. The DMC Subsystem provides the operator interface to the Mark IVa System. In addition to the software on the DMC Subsystem computers, software is required on each individual subsystem (see Fig. 3). The DMC Subsystem software consists of roughly 220,000 lines of HAL/S source code and represents an investment of almost 100 work-years.[1] The replacement of the aging Modcomp Classics will require moving that subsystem software to a new computer, and the translation of the present software to a supported computer language will require an equivalent investment. In addition, a survey of four subsystems reveals that 40 to 50 percent of the total lines of code on any subsystem are dedicated to supporting the Mark IVa Monitor and Control System (see Table 1). Any enhancement of monitor and control services to expand automation will require additional software development and modifications to DMC and all other subsystems.

An OSI-based monitor and control architecture would make extensive use of commercial off-the-shelf (COTS) software packages. All the interprocessor communication software would use OSI-based commercial software packages. Most of the subsystem software required for the Mark IVa Monitor and Control System would be replaced with these OSI-based software packages. A commercial process-control software package would be employed to support the DMC Subsystem human/machine interface. The focus of DSN resources would shift from software development and maintenance to DSN unique applications.

A. Monitor and Control Functional Requirements

The Mark IVa upgrade (1983–1985) was the first application of a local area network (LAN) in the DSN to support centralized monitor and control. The LAN provides a communication highway to all of the station subsystems. The monitor and control computers provide software for the station personnel to control the subsystems and display subsystem status information. The subsystems provide software to interpret remote commands and generate status information. Using this approach, the Mark IVa architecture distributes elements of the Monitor and Con-

[1] S. Fowler, personal communication, Data Systems Section, Jet Propulsion Laboratory, Pasadena, California, April 1991.

trol System to all of the station subsystems. The Mark IVa architecture is schematically represented in Fig. 3.

An OSI-based architecture would redistribute the monitor and control functions and concentrate the human interface in the DMC Subsystem instead of the individual subsystems being controlled. Operator directives would be processed and displays generated by the DMC Subsystem and not by the individual subsystems. An MMS network server would send configuration and control information to the subsystems and poll the subsystems for status. A real-time database would link the human interface services with the MMS network server. Figure 4 is a graphical representation of this architecture. The key element in this approach is the MMS abstract model called the Virtual Manufacturing Device (VMD), which is used to describe the externally visible characteristics of a real device. Software modules that manipulate a real device are called VMD objects. These objects can be manipulated using MMS services such as context management, variable access, domain management, semaphore management, and event management. State changes detected in the real device and defined in the VMD model can trigger MMS services. Applied to the DSN, subsystems would be modeled as one or more VMD's and operated across the network through MMS services.

The major differences between the Mark IVa architecture and an OSI-based architecture are that in the OSI-based architecture

(1) The subsystem interface is defined as a virtual machine and establishes a basis for automation through machine-to-machine communications.

(2) The architecture removes the human interface from the individual subsystems and isolates the human interface in the DMC Subsystem.

A comparison of Mark IVa functional requirements partitioned to MMS protocol services is presented in Table 2. A complete list of the 86 MMS services can be found in Appendix A.

B. Prototype Configuration

The prototype configuration incorporates three subsystems that mirror the typical functions required for station operations. This configuration is shown schematically in Fig. 5 and consists of three computers performing the following functions:

(1) The Link Monitor and Control (LMC) Subsystem provides the human interface to operate the station.

(2) The Very Long Baseline Interferometry (VLBI) Subsystem operates the Wide Channel Band (WCB) equipment for signal processing and data recording. The WCB equipment is installed at DSS 13 and is typical of DSN subsystems: uniquely designed for NASA with no commercial equivalent.

(3) The Advanced Power Meter (APM) is a collection of hardware designed to measure antenna system temperatures without continuous recalibration. This system is under development in the DSN Telecommunications Division and is installed at DSS 13.

All of the computers used to operate the prototype are Intel 80286/80386-based systems operating under Microsoft DOS 3.3 (see Table 3).

The motivation for the prototype configuration is derived from a requirement for accurate flux measurements of radio sources used for spacecraft navigation.[2] The catalog of radio sources for navigation is being expanded and the radio flux stability of a source is one of the selection criteria for the catalog. The Mark IVa architecture does not provide a mechanism for the VLBI Subsystem to control the Precision Power Meter (PPM) Subsystem; flux measurements can be acquired only through manual operation of the PPM by station personnel. The prototype was designed to examine how subsystem automation would be enhanced through the application of MMS protocols. To accomplish this goal, the prototype provides for three client–server relationships:

(1) Link Monitor and Control Subsystem (client) to the VLBI Subsystem (server).

(2) Link Monitor and Control Subsystem (client) to the Advanced Power Meter (server).

(3) VLBI Subsystem (client) to the Advanced Power Meter (server).

The ease with which these relationships are established for control and data acquisition indicates that the MMS will simplify the software development effort in the DSN.

The computers are interconnected with an ISO 8802-4 carrier band token bus local area network (5-MHz bandwidth) using Concord Communications 1210 and 1215 boards. The lower six OSI stack layers (presentation [layer six] to physical [layer one]) are downloaded to the Concord board during a configuration process. The MMS software (layer seven) was supplied by Systems Integration Specialists Corporation (SISCO). All MMS services

[2] R. Linfield and C. Jacobs, personal communication, Jet Propulsion Laboratory, Pasadena, California, February 1989.

in the prototype employ Transport Protocol Class 4 (TP4), a connection-oriented transport protocol with error detection and recovery. A commercial software package, FactoryLink by U.S. Data, was purchased and integrated into the prototype to support the human interface functions for the Link Monitor and Control Subsystem.

C. Results

The allocation of station resources is supported with the application of MMS context management services. These services include the following basic functions:

(1) Initiate.

(2) Conclude.

(3) Cancel.

(4) Abort.

In addition, the protocol specification includes the following functions to support the client–server relationship:

(1) Initiate request.

(2) Initiate indication.

(3) Initiate response (positive and negative).

(4) Initiate confirm (positive and negative).

(5) Conclude request (positive and negative).

(6) Conclude indication.

(7) Conclude response (positive and negative).

(8) Conclude confirm (positive and negative).

A typical protocol exchange is shown in Fig. 6. To begin a session, a client initiates a connection (mv_init()), which triggers an indication (u_mllp_a_assoc_ind()) on the target server. The target server responds (u_mllp_a_assoc_resp()) and triggers a confirmation (u_mv_read_conf()) on the client. This exchange is characteristic of all MMS confirmed services. A functional addressing scheme similar to that implemented in the Mark IVa System was realized through the Application Reference Name (AR-Name) conventions in the MMS protocol.

The distribution of support files (predicts) employs the MMS File Management services:

(1) Copy.

(2) Obtain.

(3) Open.

(4) Read.

(5) Close.

(6) Rename.

(7) Delete.

Again, the protocol specification includes functions to support the client–server relationship: request, indication, response, and confirmation. As in Mark IVa, the prototype Monitor and Control System distributes predict (PR) files and standards and limits (SL) files to the appropriate subsystem prior to a scheduled activity. The MMS Obtain File service is used to transmit the file names to be copied by the subsystem from the DMC Subsystem. The Obtain File Indication triggers the MMS software on the subsystem to copy the file across the LAN (see Fig. 7). The prototype code for an MMS service call requires the destination, the source file name, and the destination file name:

send_file(subsystem, source file, destination file)

or more specifically,

send_file("VLBI", "PREDICTS.DAT",

"NOVA1987.DAT")

Subsystem Directives, Displays, Events, and Alarms are supported through the application of MMS Variable Access services:

(1) Read variable.

(2) Write variable.

(3) Information report.

(4) Get variable access attributes.

(5) Define named variable.

(6) Delete variable access.

(7) Define named variable list.

(8) Get named variable list attributes.

(9) Delete named variable list.

(10) Define named type.

(11) Get named type attributes.

(12) Delete named type.

These services can be used to read or write a wide range of MMS standard variable types defined in the protocol specification:

(1) Boolean—8 bits.

(2) Integer8—8 bits.

(3) Integer16—16 bits.

(4) Integer32—32 bits.

(5) Unsigned Integer8—8 bits.

(6) Unsigned Integer16—16 bits.

(7) Unsigned Integer32—32 bits.

(8) Floating point—32 bits.

(9) Double floating point—64 bits.

(10) ASCII String8—8 bytes.

(11) ASCII String16—16 bytes.

(12) ASCII String32—32 bytes.

(13) ASCII String64—64 bytes.

(14) ASCII String128—128 bytes.

(15) ASCII String256—256 bytes.

In addition, the protocol supports the definition of complex variable types that include arrays and data structures. For example, one can define a C language data structure for a VLBI Subsystem device called the IF Distributor. This device has two inputs (character strings), two attenuation controls (integer values), and two total power integrators (floating-point values):

```
struct IFD_TYPE { char  if_1_in[4],
                  char if_2_in[4],
                  short if_1_att,
                  short if_2_att,
                  float if_1_pwr,
                  float if_2_pwr } ifd_status
```

When defined as an MMS complex variable, the entire structure can be written or read across the network as though it were a single variable with the name "ifd_status" of type "IFD_TYPE." Arrays of simple variables and structures can be handled in the same way. In addition, structures can be nested, that is, a structure can be within a structure.

In an MMS-based Monitor and Control System, operator directives are entered, converted to data, and transmitted by the DMC Subsystem to the individual subsystems using the confirmed MMS Variable Write service. An MMS confirmed service requires a response from the server to the client to verify the success or failure of the service. For example, the VLBI Wide Channel Band Subsystem has an attenuation control that requires operator adjustment. In the prototype system, a change of attenuation to 23 dB is entered through the graphical user interface and the MMS Variable Write service is used to transmit the data across the LAN to the VLBI Subsystem. The prototype code for the MMS Variable Write service requires the subsystem destination, the variable name, the variable type, and the data:

```
write_named_var("VLBI", "if_1_att", "Integer16", 23)
```

Operator displays are constructed on the DMC Subsystem from data obtained across the LAN using the confirmed MMS Variable Read service. In the prototype, a polling system was implemented to read the data from each subsystem on a periodic basis. Again based on the earlier VLBI example, the prototype code for the MMS Variable Read service requires the subsystem source, the variable name and the variable type:

```
read_named_var("VLBI","if_1_att", "Integer16")
```

Event and Alarm conditions are reported across the LAN using the unconfirmed MMS Information Report (info_report) service, a variation on the Variable Write service. Unlike the Variable Read and Write services, which must be initiated by the client, the Information Report service can be initiated by a server. The protocol specification does not require the client to acknowledge an information report, therefore the service is unconfirmed. Though unconfirmed (see Fig. 8) at the application layer, the Information Report service employs TP4, a reliable transport protocol service. The first goal in the prototype effort was to simulate the functions in DSN operations today. The Mark IVa Event/Alarm messages consist of text information for the operator. The messages are transmitted by the subsystems to the DMC Subsystem with a category identification:

(1) PROMPT.

(2) PROGRESS advisory.

(3) COMPLETION advisory.

(4) DEVIATION advisory.

(5) WARNING alarm.

(6) CRITICAL alarm.

(7) EMERGENCY alarm.

In the prototype, seven MMS-named variables are defined with the same Mark IVa names, each as a 64-character ASCII string. The prototype code for the DSN Event Message service requires the subsystem destination, the event name, and the message:

event_msg(subsystem, type, message)

or more specifically,

event_msg("VLBI", "COMPLETION",
"Data recording started")

The commercial monitor and control software package employed in the prototype offers another approach to reporting event and alarm conditions. Commercial monitor and control packages provide a service to generate and manage alarms based on changes detected in their real-time database. Again using the VLBI Subsystem as an example, the VLBI Subsystem controller monitors all of its devices every 15 sec. If a deviation in the expected configuration is detected, the VLBI Subsystem software can use the MMS Information Report service to update the DMC Subsystem:

dsn_send_info_rpt(subsystem, var_name, var_type, data)

or more specifically,

dsn_send_info_rpt("VLBI", "if_1_att", "Integer16", 18)

The MMS network server on the DMC Subsystem updates the real-time database and the commercial alarm-management software triggers an operator alarm.

In the Mark IVa era, monitor data blocks are transmitted from each subsystem to the DMC Subsystem. In turn, the DMC Subsystem acts as a middleman and redistributes monitor data to all of the subsystems. The data contained in the monitor data block are individually negotiated in advance, in detail, down to the bit level. In an MMS-based architecture, the data acquired through Variable Read services to support DMC Subsystem displays replace the current monitor data blocks. The redistribution of monitor data is supported with Variable Write services. In addition, the careful application of direct subsystem-to-subsystem Variable Read services would eliminate the middleman function of the DMC Subsystem.

D. Prototype Performance

The performance of any computer system is highly dependent on the specific implementation and is not a function of the protocol alone. Some of the factors that impact system performance are

(1) Software architecture.

(2) Software implementation.

(3) Operating systems.

(4) Computer hardware.

(5) Communication medium.

(6) Supporting hardware.

The resource constraints for this research effort defined the hardware and software options available for the prototype. The selection of the token bus LAN was one consequence of the constraints. In addition, the effort did not permit the acquisition of a token bus network analyzer, the lack of which limited the range of performance tests. Given these limitations, performance tests were developed to provide a baseline for comparison with future implementations. These tests focus on the key services:

(1) Time required to establish a connection.

(2) File transfer data rates.

(3) Variable write data rates.

(3) Variable read data rates.

The average time required to establish a connection on the prototype system ranged from 0.2 to 0.3 sec. This measurement includes the time to build, transmit, and process all four elements of the specified protocol shown in Fig. 6. A direct comparison with the present system is difficult because today's DSN is based on connectionless protocols. However, the time required for an operator directive to be acknowledged in the present system ranges from 1 to 3 sec.

The performance results of the MMS Obtain File service are plotted in Fig. 9. These results indicate that rates of 4500 bytes/sec can easily be achieved. The difference in the performance of the two computers (8 MHz versus 6 MHz) reflects not only the central processing unit (CPU) performance but the difference in disk access speed of the two computers. The average disk access speed for the Everex (8-MHz) computer was 19.9 msec while the IBM AT (6-MHz) computer disk access speed was 37.9 msec. The Mark IVa support files range in size from 1000 to 80,000 bytes; the DMC Subsystem transmits these files at a rate of 1800 bytes/sec. Based on the prototype results, the implementation of MMS File Management services would double the present throughput performance.

The performance results for the Variable Access services are plotted in Figs. 10 and 11, and reflect their sensitivity to the CPU and memory speed of the prototype computers. The data rates reported for the prototype tests

are for application data. The number of bytes transmitted in an MMS message packet is a function of the specific MMS service and the application data. For example, the packet generated to write a single variable string of 1000 characters contains 1019 bytes. The packet generated to write an array of 500 16-bit integers (1000 bytes of data) contains 1948 bytes. The application layer data included to support the processing of the packet accounts for the difference in the packet size. (See Appendix B for a detailed example of the MMS message formulation.) Arrays of 16-bit integers, arrays of 32-bit floating point variables, and a series of ASCII strings were created to generate tests with packets of 4, 20, 200, 464, 732, and 1000 bytes of application data.

Once viewed as unnecessary protocol overhead, the application layer data perform a vital role in the communication process by defining all elements of the data packet. The benefits are realized through application software that is independent of the communication process. In the past, protocol design was motivated by communication bandwidth limitations. Today, modern LAN's have alleviated many of the bandwidth limitations. The performance results obtained for MMS Variable Access services are typical for packet transfer protocols. Note that Variable Reads and Writes of individual variables are less efficient than Reads and Writes of large data structures and arrays. Based on these results, an MMS-based Monitor and Control System should require subsystem VMD's designed to use data structures and arrays whenever possible and appropriate.

E. Intersubsystem Automation

One of the primary objectives of the prototype effort was the exploration of subsystem-to-subsystem automation using MMS protocols. For this effort, the software for the Advanced Power Meter was designed to operate in two modes:

(1) In single measurement mode, the power meter performs a simulated discrete system temperature measurement.

(2) In continuous mode, the power meter performs repeated system temperature measurements.

Software was developed for the VLBI Subsystem controller to establish a direct client–server relationship with the power meter controller. Under the command of the DMC Subsystem, the VLBI Subsystem can set the power meter operating mode (see Fig. 12). In single measurement mode, VLBI uses the MMS Variable Write service to trigger the power meter to perform a single system temperature measurement based on the contents of the VLBI predict schedule. On completion of the measurement, the power meter reports the results back to VLBI using the MMS Information Report service. In continuous mode, VLBI uses the MMS Variable Write service to set the power meter to continuous sampling mode, and the VLBI Subsystem performs an MMS Variable Read to obtain the latest results based on its predict schedule. The VLBI Subsystem and DMC Subsystem interfaces to the power meter are identical. The establishment of this relationship was straightforward and simple to implement.

F. Commercial Monitor and Control Packages

One of the benefits of implementing systems based on the standards is the availability of products designed to operate with the standards. In the process control world, there are a number of commercial products available for factory automation. These products provide graphical user interfaces for monitor and control, communication services to process control devices, logging of real-time performance data, operator alarm notification, and a variety of other services. The products are table driven and designed to be tailored and installed in any process control environment without software modifications. Several companies which produce these products have developed MMS interfaces to their systems. The U.S. Data product FactoryLink is one example. The FactoryLink DOS product (for IBM PC's) has been integrated in the prototype to provide operator control and a graphical user interface with time-ordered (trending) plots of real-time data [19].

G. Future Steps

The MMS prototype is the basis for a new Monitor and Control System to be developed and installed at DSS 13 in 1992. This system will provide centralized monitor and control for all of the core subsystems planned for the new 34-meter beam waveguide antenna. The DSS-13 effort will provide an expanded test bed to examine other MMS services such as

(1) Domain management.

(2) Semaphore management.

(3) Event management.

(4) Journal management.

Beyond the MMS effort, a full and complete evaluation of FTAM services and protocols is necessary before FTAM can be applied to centralized data recording at the DSN stations.

VI. Conclusions

The DSN can adopt an OSI architecture and would benefit from the application of OSI services and protocols in several areas.

(1) OSI would provide the DSN with a set of interprocessor communication standards that can be specified in all future implementations. Wide industrial support for OSI will insure a selection of vendors while providing compatibility with future implementations.

(2) The development costs for OSI products will be distributed over the worldwide market, reducing DSN costs for network services. Manufacturers will compete to have highly reliable, high performance products at a relatively low price.

(3) The application of the MMS would provide the foundation for increased automation of the DSN. MMS establishes the client–server relationship and services required to operate multiple computers operating as a single system.

(4) The application of FTAM services would provide the basis for network file servers in the DSN. FTAM services would be used to record data, play back data, and transfer files more reliably.

(5) The application of standard OSI protocols establishes an internetworking system based on the ISO 8473 internetwork protocol. Commercial Wide Area Network (WAN) bridges and routers would be employed to interconnect the distributed DSN facilities at reduced investment. In addition, the utilization of the ISO internetwork protocol would simplify the exchange of data between agencies, particularly as the international community moves to OSI.

(6) The application of OSI Network Management would provide the services to control communication resources, identify fault conditions, account for network utilization, and insure full security in an open system.

New products based on the OSI services are already under development. A number of companies are developing programmable logic controllers (PLC's) based on the OSI MMS standards; other companies are developing MMS servers to support commercial database products. Commercial products are already available to provide monitor and control systems for factories and their adaptation to OSI services is under way.

The cost of transition to OSI may be equal to, or even exceed, the cost of other networking solutions in the short term. However, the federal policy to adopt OSI is based on the long-term cost benefits (estimated by the National Research Council at 30 to 80 percent of the implementation cost for new computer systems) [6]. OSI-based MMS products have been introduced by companies such as Digital Equipment Corporation (DEC), International Business Machines (IBM), Hewlett-Packard, and Motorola for a wide range of computers, indicating industry's commitment to the standards. The cost of software development, test, and integration is thus distributed over their large customer base. In addition, the MMS Standard has been developed to meet the diverse demands of commercial industry and offers a spectrum of services that this research has shown to be more than adequate to meet the DSN requirements for DSCC monitor and control in the next century. Moreover, the layered OSI architecture would enhance the DSN's ability to cope with changing requirements and technologies.

In a broader sense, the adoption of ISO protocols would benefit the DSN by incorporating it into the world networking community. The Consultative Committee for Space Data Systems [2] has adopted the OSI architecture as the basis for international space data systems and the ISO 8473 network protocol is a key component of the CCSDS architecture. In addition, GOSIP requires all federal procurement of networking services to employ ISO protocols [1]. In response to GOSIP, the NASA Science Internet (NSI) is making the transition to OSI and NASCOM is planning a transition to OSI in the 1990s [18].[3] The adoption of an OSI architecture and the application of OSI protocols is necessary to meet the demands of DSN customers in a cost-effective manner.

[3] R. Nitzen, personal communication, NASA Headquarters, Washington, DC, April 1990.

References

[1] U.S. Department of Commerce, National Bureau of Standards, *Federal Information Processing Standards Publication—146, Government Open Systems Interconnection Profile*, Washington, DC, August 1988.

[2] Consultative Committee for Space Data Systems (CCSDS), *Advanced Orbiting Systems, Network and Data Links: Architectural Specification*, Blue Book, CCSDS 701.0-B-1, CCSDS Secretariat, NASA, Washington, DC, October 1989.

[3] C. Morgan, "Department of Defense Plans for Open Systems Interconnection," in *Case Studies in Implementing OSI* (tutorial), Interop Incorporated, Mountain View, California, October 1989.

[4] U.S. Department of Defense, *Military Standard Transmission Control Protocol*, MIL-STD-1776, Washington, DC, August 1983.

[5] U.S. Department of Defense, *Military Standard Internet Protocol*, MIL-STD-1777, Washington, DC, August 1983.

[6] National Research Council, *Transport Protocols for Department of Defense Data Networks*, PB85-176147, National Technical Information Service, Springfield, Virginia, February 1985.

[7] U.S. Department of Defense, *The Department of Defense Open Systems Interconnection (OSI) Implementation Strategy*, planning document, Reston, Virginia, May 1988.

[8] United States Congress, "Computer Security Act of 1987," Public Law 100-235, *Congressional Record*, vol. 133 (1987), approved January 8, 1988.

[9] R. C. Brooks, "Department of Veterans Affairs," in *Case Studies in Implementing OSI* (tutorial), Interop Incorporated, Mountain View, California, October 1989.

[10] International Organization for Standardization, *Information Processing Systems—Open Systems Interconnection—Basic Reference Model*, ISO 7498, American National Standards Institute, New York, October 1984.

[11] J. Henshall and S. Shaw, *OSI Explained, End-to-End Computer Communication Standards*, West Sussex, England: Ellis-Horwood Limited, pp. 10–20, 1988.

[12] K. G. Knightson, J. Larmouth, and T. Knowles, *Standards for Open Systems Interconnection*, New York: McGraw-Hill Book Company, pp. 12–19, 20–54, 259–301, 1988.

[13] International Organization for Standardization, *Information Processing Systems—Open Systems Interconnection—Manufacturing Message Specification*, ISO 9506, American National Standards Institute, New York, 1989.

[14] International Organization for Standardization, *Information Processing Systems—Open Systems Interconnection—Basic Reference Model, Part 4: Management Framework*, ISO 7498-4 (Draft International Standard), American National Standards Institute, New York, 1988.

[15] International Organization for Standardization, *Information Processing Systems—Open Systems Interconnection—Systems Management Overview*, ISO 10164 (Draft Proposal), American National Standards Institute, New York, 1990.

[16] International Organization for Standardization, *Information Processing Systems—Open Systems Interconnection—Structure of Management Information*,

ISO 10165 (Draft Proposal), American National Standards Institute, New York, 1990.

[17] R. Brennan, K. Thompson, and R. Wilder, "Mapping the X Window onto Open Systems Interconnection Standards," *IEEE Network Magazine*, vol. 4, no. 2, pp. 32–40, May 1991.

[18] *NASCOM Service Gateway Protocol Study*, Computer Sciences Corporation, Greenbelt, Maryland, July 1988.

[19] United States Data Corporation, *FactoryLink Software System*, Publication TP-FLDOS 1/90, Richardson, Texas, January 1990.

[20] P. Norton, *The Norton Utilities Version 5.0 User's Guide*, Peter Norton Computing, Inc., Santa Monica, California, pp. 266–270, 1990.

Table 1. Subsystem lines of code supporting the Mark IVa Monitor and Control System.

Subsystem	Lines of code	Percentage of code for M&C
Monitor and Control	220,000	100
VLBI	107,152	40
Command	23,644	40
Telemetry	60,940	50
Radio Science	53,335	40

Table 2. The functional requirements for Mark IVa Monitor and Control (on the left) and the equivalent MMS protocol services (on the right).

Mark IVa functional requirements	MMS protocol services
Resource allocation	Context Management
Supported with the distribution of Functional Address Tables by Complex Monitor and Control. These tables are used to identify subsystems engaged to support specific station activities. All communications between subsystems are based on addresses in these tables.	Supported with context (connection) management services through the Application Reference Name.
Distribution of support data	File Management
Support data are received by the DMC Subsystem from the Network Operation Control Center and redistributed to individual subsystems prior to a station activity. Subsystems that require support data must have software to accept and process support data blocks distributed by the DMC Subsystem.	Support data files are transmitted to and from the DMC Subsystem using the FTAM services provided in the MMS specification: File Copy, File Obtain, File Open, File Read, File Close.
Operator directives	Variable Access
Operator directives to configure or control the subsystems are entered by an operator at DSCC Monitor and Control (DMC) and processed into messages which are transmitted to the appropriate subsystem across the LAN. Subsystems process directive messages received from DMC to perform the requested functions.	Operator directives for subsystem configuration and control are entered on the DMC Subsystem, converted to the data required by the subsystem, and MMS Variable Write services deliver the data across the network to the subsystem.
Subsystem displays	Variable Access
Subsystem health and performance information is reported to the operator through subsystem displays. Subsystem display data blocks are generated and transmitted across the LAN by each subsystem on request from an operator. The DMC Subsystem is responsible for processing and presenting of subsystem displays for the operator.	Subsystem data for health and performance are obtained using MMS Variable Read services. The data are presented to the operator through graphical displays built by the DMC Subsystem.
Events and Alarms	Variable Access
Subsystem Event and Alarm conditions are generated and reported to DMC. The DMC Subsystem processes Event messages, displays the messages to the operator and logs the messages to an archive file.	Subsystem Event or Alarm conditions are reported to the DMC Subsystem using the MMS Information Report service. MMS Journal services log Events and Alarms to an archive file.
Monitor data	Variable Access
Monitor data are transmitted by each active subsystem to DMC based on negotiated interface agreements. In addition, the DMC Subsystem redistributes monitor data to all of the subsystems based on the negotiated interface agreements.	Monitor data are obtained using the MMS Variable Read services. The redistribution of monitor data is accomplished with MMS Variable Write services.

Table 3. The computer hardware used for the prototype is listed with its Norton Performance Index [20].

Subsystem	Computer	Central processing unit	Clock speed, MHz	Norton Performance Index
Link Monitor and Control (LMC)	BiLink 386	80386	20	23.0
Very Long Baseline Interferometry (VLBI)	Everex 1800	80286	8	7.7
Advanced Power Meter (APM)	IBM AT	80286	8	5.7

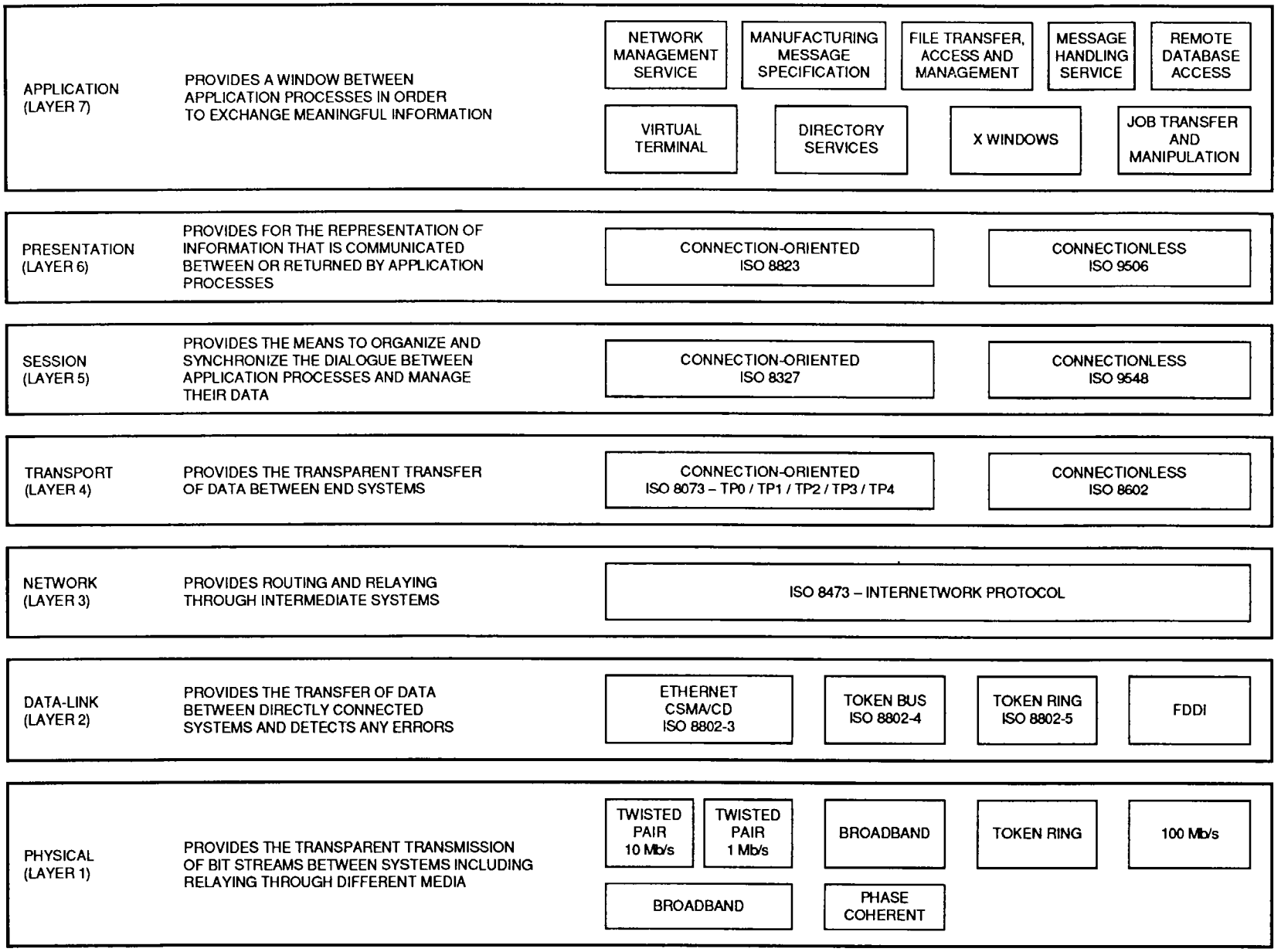

Fig. 1. Open Systems Interconnection architecture provides a wide range of services and protocols to support the growing demands on distributed systems.

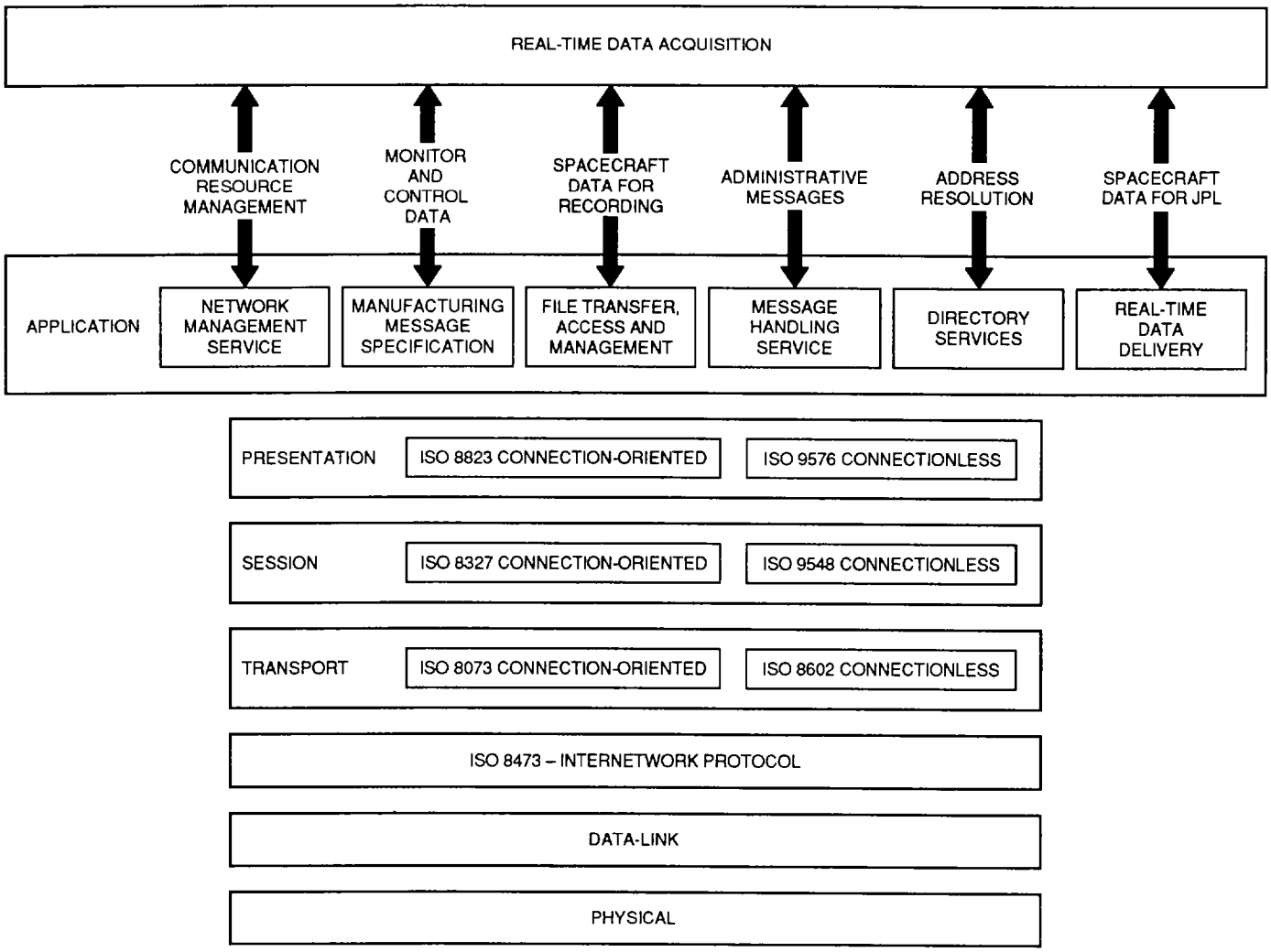

Fig. 2. An Open Systems Interconnection architecture will access communication services through application entities.

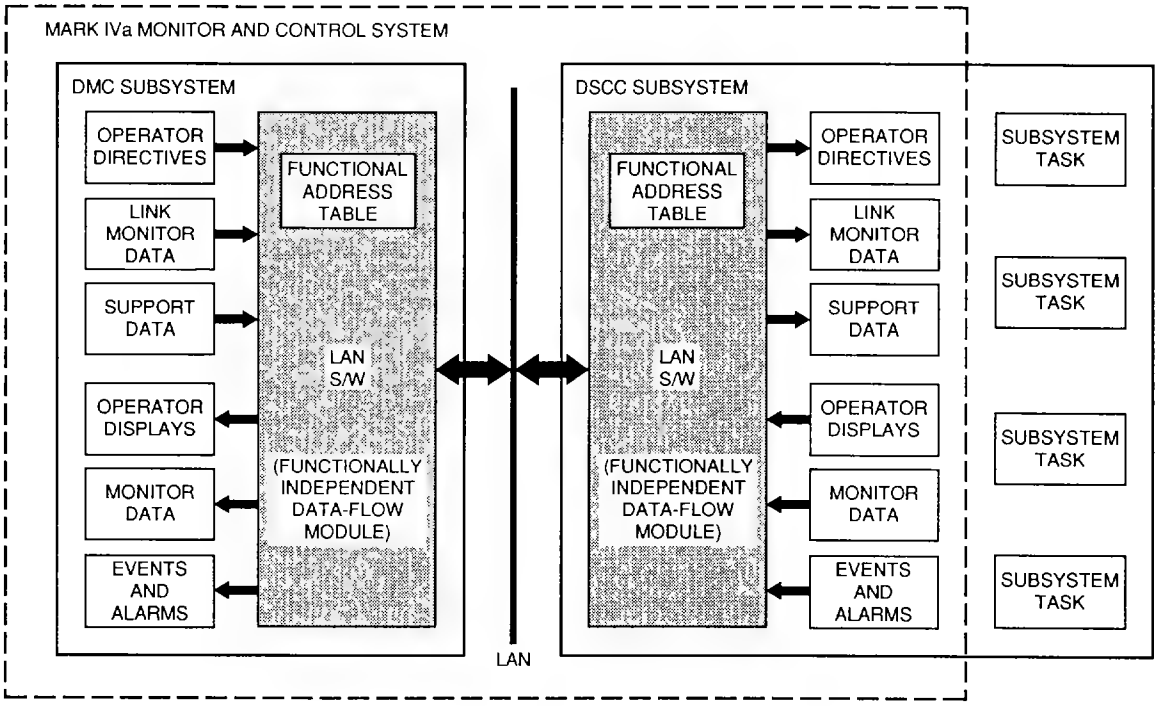

Fig. 3. A schematic representation of the Mark IVa monitor and control architecture, which requires software on each subsystem to support monitor and control.

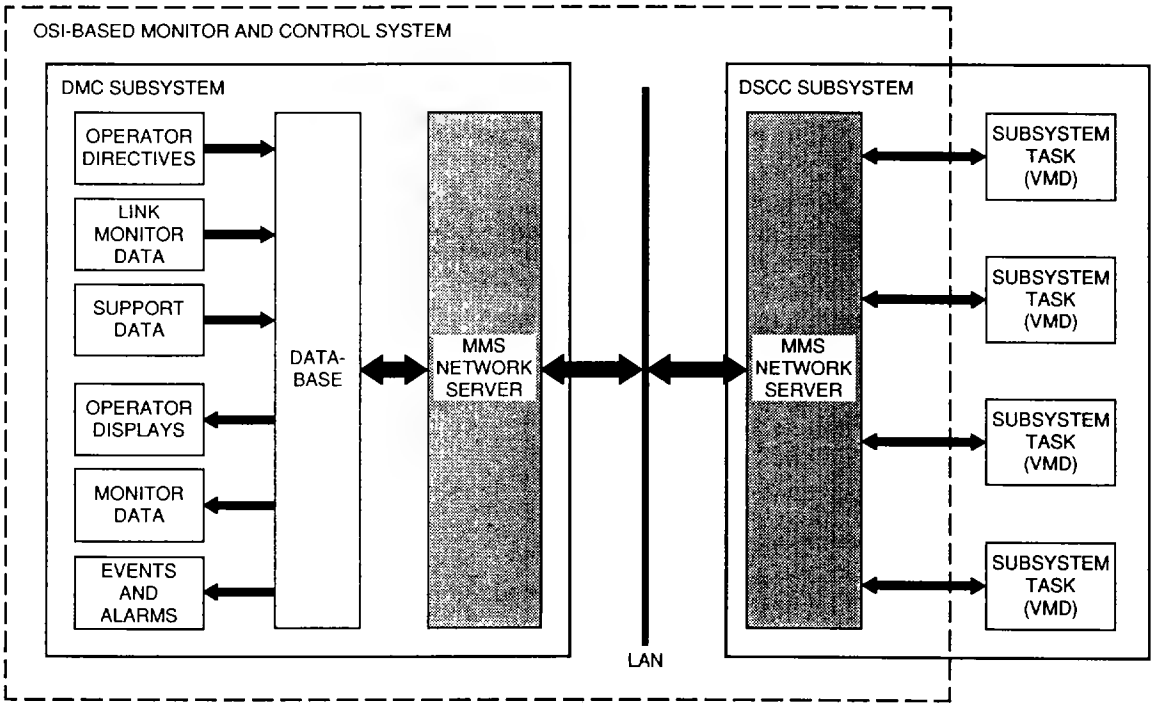

Fig. 4. A schematic representation of an OSI-based monitor and control architecture, which employs MMS to support all monitor and control functions.

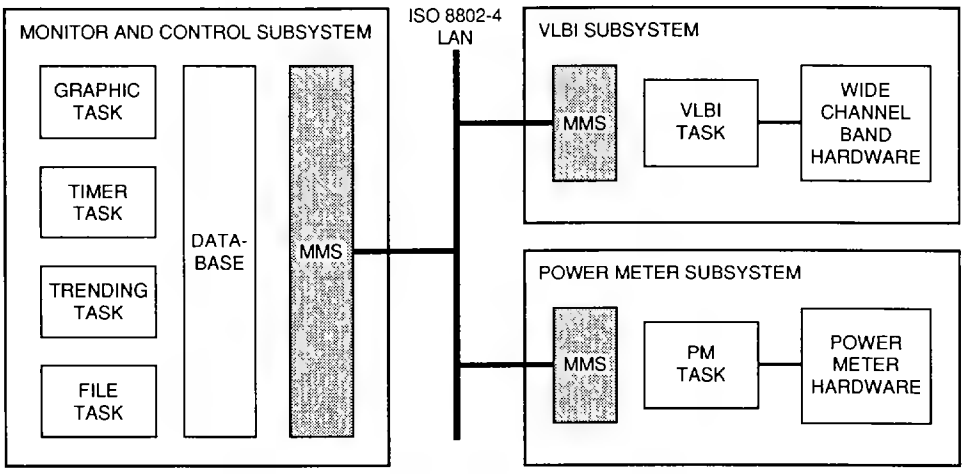

Fig. 5. The OSI monitor and control prototype consists of three subsystems, Link Monitor and Control, VLBI, and the Advanced Power Meter.

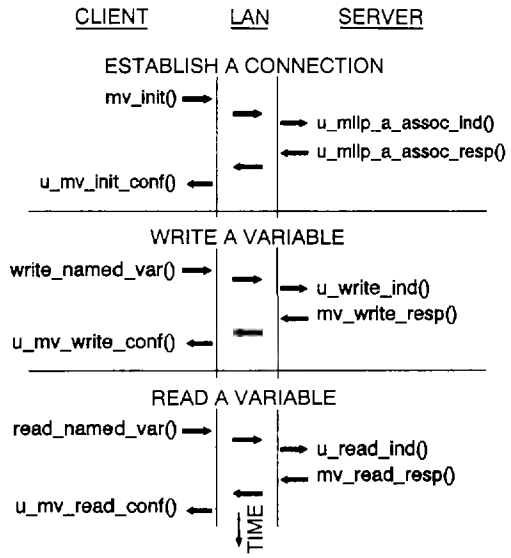

Fig. 6. The protocol exchange between client and server is shown for Context Management and Variable Access services as specified in the MMS Standard.

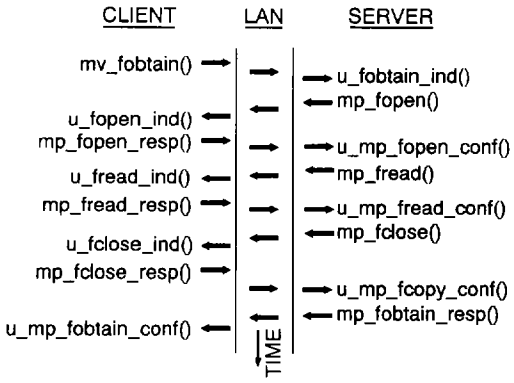

Fig. 7. The File Transfer, Access and Management protocol exchange for the Obtain File service is outlined above as specified in the MMS Standard.

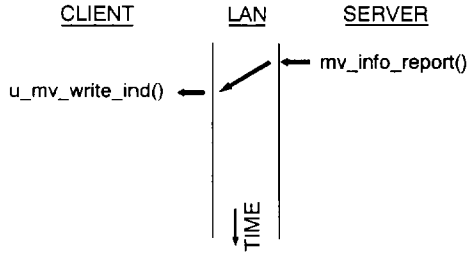

Fig. 8. The protocol exchange for the Information Report service as specified in the MMS Standard.

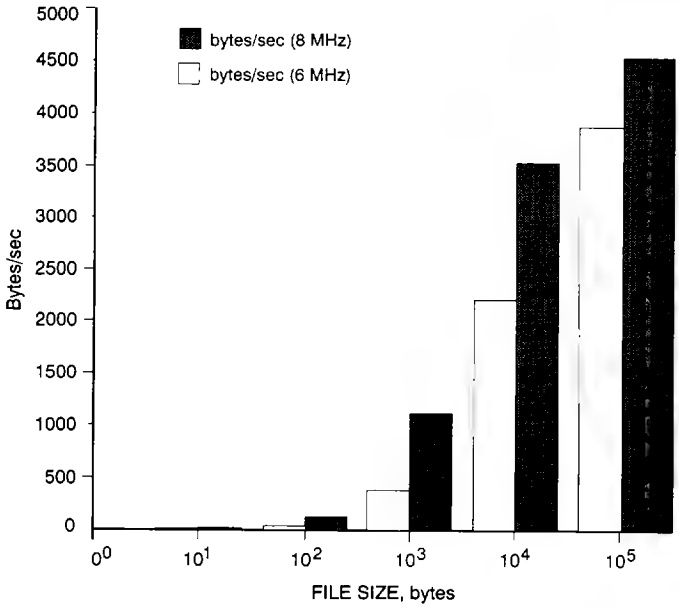

Fig. 9. The performance of the MMS Obtain File service in the prototype environment.

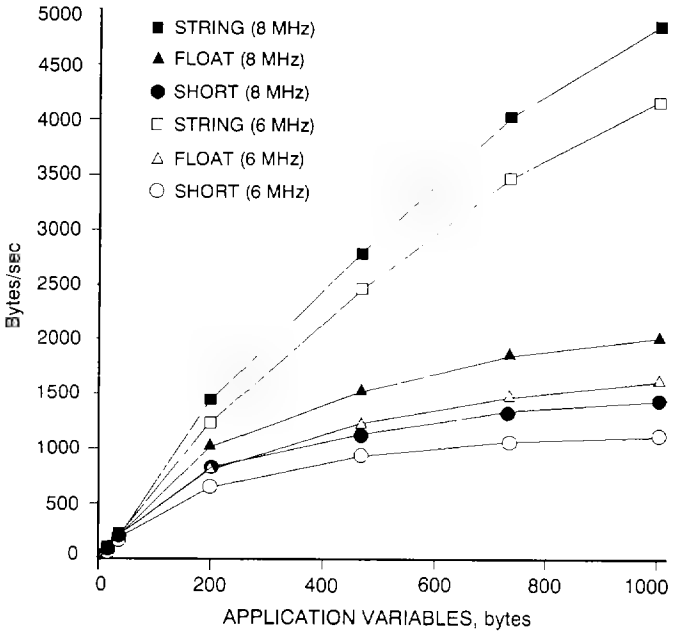

Fig. 10. The performance of the MMS Variable Write service in the prototype environment for application data in bytes/sec.

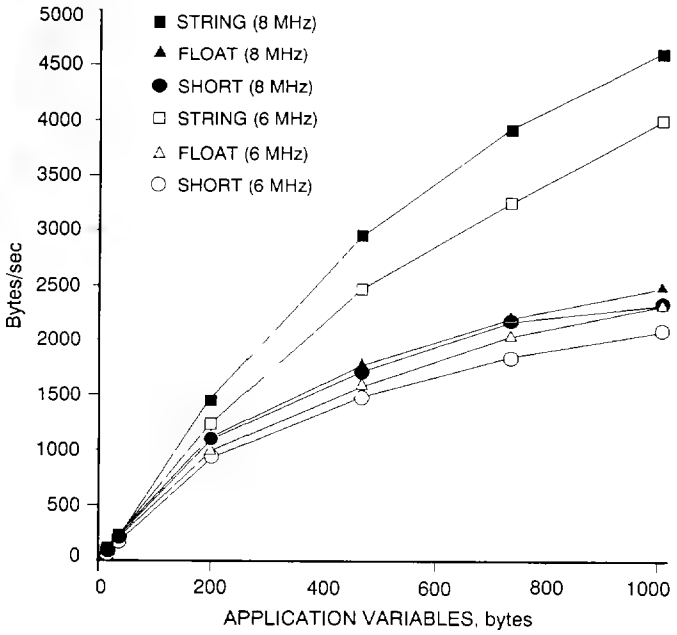

Fig. 11. The performance of the MMS Variable Read service in the prototype environment for application data in bytes/sec.

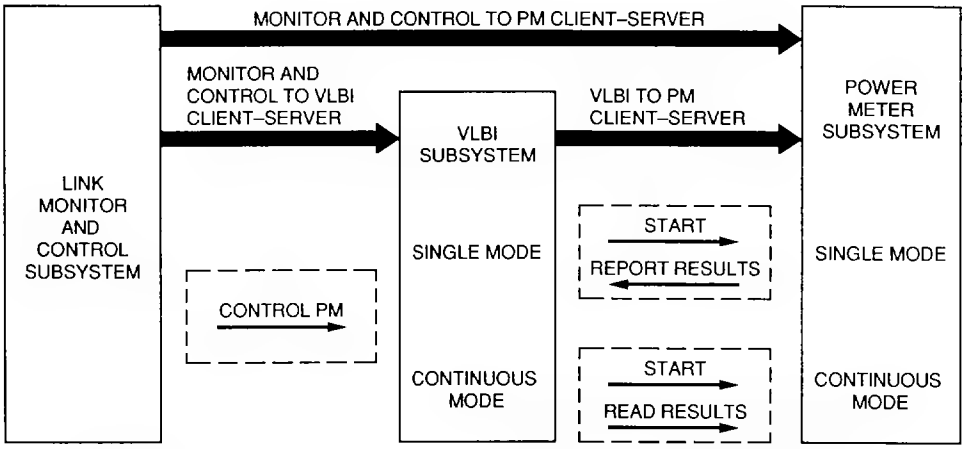

Fig. 12. The client–server relations for automation of the Advanced Power Meter in the prototype Monitor and Control Subsystem.

Appendix A

List of MMS Protocol Services

I. Connection Services

Initiate

Conclude

Cancel

Abort

II. VMD Support Services

Status

Unsolicited Status

GetNameList

Identify

Rename

GetCapabilityList

III. Domain Management Services

InitiateDownSequence

DownloadSegment

TerminateDownSequence

InitiateUploadSequence

UploadSegment

TerminateUploadSequence

RequestDomainDownload

RequestDomainUpload

LoadDomainContent

StoreDomainContent

DeleteDomain

GetDomainAttributes

IV. Program Invocation Management Services

CreateProgramInvocation

DeleteProgramInvocation

Start

Stop

Resume

Reset

Kill

GetProgramInvocationAttributes

V. Variable Access Services

Read

Write

InformationReport

GetVariableAccessAttributes

DefineNamedVariable

DefineScatteredAccess

GetScatteredAccessAttributes

DeleteVariableAccess

DefineNamedVariableList

GetNamedVariableListAttributes

DeleteNamedVariableList

DefineNamedType

GetNamedTypeAttributes

DeleteNamedType

VI. Semaphore Management Services

TakeControl

RelinquishControl

DefineSemaphore

DeleteSemaphore

ReportSemaphoreStatus

ReportPoolSemaphoreStatus

ReportSemaphoreEntryStatus

AttachToSemaphoreModifier

VII. Operator Communication Services

Input

Output

VIII. Event Management Services

DefineEventCondition

DeleteEventCondition

GetEventConditionAttributes

ReportEventConditionStatus

AlterEventConditionMonitoring

TriggerEvent

DefineEventAction

DeleteEventAction

GetEventActionAttributes

ReportEventActionStatus

DefineEventEnrollment

DeleteEventEnrollment

GetEventEnrollmentAttributes

ReportEventEnrollmentStatus

AlterEventEnrollment

EventNotification

AcknowledgeEventEnrollment

GetAlarmSummary

GetAlarmEnrollmentSummary

AttachToEventConditionModifier

IX. Journal Management Services

ReadJournal

WriteJournal

InitializeJournal

ReportJournalStatus

CreateJournal

DeleteJournal

X. File Management Services

ObtainFile

FileOpen

FileRead

FileClose

FileRename

FileDelete

FileDirectory

Appendix B

MMS Message Formulation

The precise formulation of an MMS message is dependent on the service and defined in the standard specification. The construction and interpretation of MMS messages in the prototype were performed by commercial MMS software, and detailed knowledge of the process is not required to employ the protocol. However, a brief examination of how one MMS message is defined and constructed will yield insight into the complexity and versatility of the protocol.

The process for the MMS Variable Write service begins with an MMS protocol data unit (PDU):

```
MMSpdu    ::  {
        confirmed-RequestPDU            [0]   IMPLICIT    Confirmed-RequestPDU,
        confirmed-ResponsePDU           [1]   IMPLICIT    Confirmed-ResponsePDU,
        confirmed-ErrorPDU              [2]   IMPLICIT    Confirmed-ErrorPDU,
        unconfirmed-PDU                 [3]   IMPLICIT    Unconfirmed-PDU,
        rejectPDU                       [4]   IMPLICIT    RejectPDU,
        cancel-Request-PDU              [5]   IMPLICIT    Cancel-RequestPDU,
        cancel-Response-PDU             [6]   IMPLICIT    Cancel-ResponsePDU,
        cancel-ErrorPDU                 [7]   IMPLICIT    Cancel-ErrorPDU,
        initiate-RequestPDU             [8]   IMPLICIT    Initiate-RequestPDU,
        initiate-ResponsePDU            [9]   IMPLICIT    Initiate-ResponsePDU,
        initiate-ErrorPDU              [10]   IMPLICIT    Initiate-ErrorPDU,
        conclude-RequestPDU            [11]   IMPLICIT    Conclude-RequestPDU,
        conclude-ResponsePDU           [12]   IMPLICIT    Conclude-ResponsePDU,
        conclude-ErrorPDU              [13]   IMPLICIT    Conclude-ErrorPDU
}
```

The confirmed-RequestPDU follows with the Write-Request:

```
Confirmed-RequestPDU          ::=   SEQUENCE     {
        invokeID     Unsigned32,
        listOfModifier       SEQUENCE     OF   Modifier     OPTIONAL,
        ConfirmedServiceRequest,
        [79]   CS-Request-Detail         OPTIONAL
        --  shall   not   be   transmitted      if   value    is   NULL
}

Write-Request         ::=   SEQUENCE     {
      variableAccessSpecification              VariableAccessSpecification,
       listOfData                                [0]   IMPLICIT     SEQUENCE     OF   DATA
      }
```

The variable access specification and the data follow:

```
VariableAccessSpecification          ::  CHOICE    {
        listOfVariable              [0]   IMPLICIT     SEQUENCE     OF  SEQUENCE     {
                variableSpecification           VariableSpecification,
                alternateAccess                 [5]   IMPLICIT     AlternateAccess       OPTIONAL
                },
        variableListName         [1]   ObjectName
        }
```

Each variable name, address and description is included in the PDU:

```
VariableSpecification         ::=  CHOICE    {
    name                                [0]   ObjectName,
    address                             [1]   Address,
    variableDescription                 [2]   IMPLICIT     SEQUENCE     {
        address                     Address,
        typeSpecification           TypeSpecification
        },
    scatteredAccessDescription          [3]   IMPLICIT     ScatteredAccessDescription,
    invalidated                         [4]   IMPLICIT     NULL
    }
```

The name is presented in the form of ObjectName:

```
ObjectName       ::=  CHOICE    {
        vmd-specific            [0]   IMPLICIT     Identifier,
        domain-specific         [1]   IMPLICIT     SEQUENCE     {
                domainID        Identifier,
                itemID          Identifier
                },
        aa-specific             [2]   IMPLICIT     Identifier
        }
```

The address parameter is presented in the form of Address:

```
Address      ::=  CHOICE    {
        numericAddress                  [0]   IMPLICIT      Unsigned32,
        symbolicAddress                 [1]   IMPLICIT      VisibleString,
        unconstrainedAddress            [2]    IMPLICIT     OCTET    STRING
        }
```

And the type specification takes the following form:

```
TypeSpecification        ::=  CHOICE   {
        typeName         [0]  ObjectName,
        array            [1]  IMPLICIT    SEQUENCE    {
                packed                  [0]  IMPLICIT    BOOLEAN    DEFAULT    FALSE,
                numberOfElement         [1]  IMPLICIT    Unsign32,
                elementType             [2]  TypeSpecification
                },
        structure            [2]  IMPLICIT    SEQUENCE    {
                packed                  [0]  IMPLICIT    BOOLEAN    DEFAULT    FALSE,
                components              [1]  IMPLICIT    SEQUENCE    OF  SEQUENCE    {
                  componentName          [0]  IMPLICIT    Identifier     OPTIONAL,
                componentType           [1]  TypeSpecification
                  }
              },
        --  Simple
        boolean              [3]  IMPLICIT    NULL,             --  BOOLEAN
        bit-string           [4]  IMPLICIT    Integer32,        --  BIT-STRING
        integer              [5]  IMPLICIT    Unsigned8,        --  INTEGER
        unsigned             [6]  IMPLICIT    Unsigned8,        --  UNSIGNED
        floating-point       [7]  IMPLICIT    SEQUENCE    {
                format-width          Unsigned8,       --  number   of  bits   in
                                                       --  fraction    plus   sign
             exponent-width        Unsigned8          --  size   of  exponent    in  bits
             },
        real                 [8]  IMPLICIT    SEQUENCE    {
                base            [0]  IMPLICIT    INTEGER(2|10),
                exponent        [1]  IMPLICIT    INTEGER,      --  max   number    of  octets
                mantissa        [2]  IMPLICIT    INTEGER       --  max   number    of  octets     },
          octet-string                [9]  IMPLICIT    Integer32,       --  OCTET-STRING
          visible-string              [10]  IMPLICIT    Integer32,      --  VISIBLE-STRING
          generalized-time            [11]  IMPLICIT    NULL,           --  GENERALIZEDTIME
          binary-time                 [12]  IMPLICIT    BOOLEAN,        --  BINARY-TIME
          bcd                         [13]  IMPLICIT    Unsigned8       --  BCD
      }
```

Finally, the data are specified:

```
Data    ::=  CHOICE   {
     --  context    tag  0  is  reserved    for  AccessResult
     array                    [1]    IMPLICIT    SEQUENCE    OF  Data,
     structure                [2]    IMPLICIT    SEQUENCE    OF  Data,
     boolean                  [3]    IMPLICIT    BOOLEAN,
     bit-string               [4]    IMPLICIT    BIT  STRING,
     integer                  [5]    IMPLICIT    INTEGER,
     unsigned                 [6]    IMPLICIT    INTEGER,
     array                    [7]    IMPLICIT    FloatingPoint,
     real                     [8]    IMPLICIT    REAL,
     octet-string             [9]    IMPLICIT    OCTET   STRING,
     visible-string          [10]    IMPLICIT    VisibleString,
     generalized-time        [11]    IMPLICIT    GeneralizedTime,
     binary-time             [12]    IMPLICIT    TimeOfDay,
     bcd                     [13]    IMPLICIT    INTEGER,
     booleanArray            [14]    IMPLICIT    BIT  STRING
     }
```

Once formulated into a PDU, the message is encoded in Abstract Syntax Notation One (ASN.1), then passed to the presentation layer and down through the lower layers of the protocol stack. At each step along the way, a protocol data unit is added by each layer until the entire message is formulated and transmitted on the physical network.